T02 Solution

TODO #1:

```
%%time
dask_arr = da.random.normal(20, 0.1, size=(20000, 20000),
chunks=(10000,10000))
dask_arr_mean = dask_arr.mean(axis=0).compute()
```

TODO #2:

```
%%time
dask_arr = da.random.normal(20, 0.1, size=(20000, 20000),
chunks=(30,30))
dask_arr_mean = dask_arr.mean(axis=0).compute()
```

TODO #3:

```
dask_arr_sum = dask_arr + dask_arr.T
result = dask_arr_sum.mean(axis=1)
result.compute()
```

TODO #4:

```
The first dimension will remain unchanged in size.
```

TODO #5:

```
#TODO 5: Investigate how many flights were taken that were not
canceled.
len(ddf[~ddf['Cancelled']].compute())
```

TODO #6:

```
#TODO 6: Show the count of non-canceled flights originating from each
airport.
ddf[~ddf['Cancelled']].groupby("Origin").Origin.count().compute()
```

T02 Solution

TODO #7:

```
#TODO 7: Calculate the average departure delay for each airport.
ddf.groupby("Origin").DepDelay.mean().compute()
```

TODO #8:

```
#TODO 8: Determine which day of the week experiences the highest
average departure delay.
ddf.groupby("DayOfWeek").DepDelay.mean().idxmax().compute()
```

TODO #9:

```
#TODO 9: Given that the distance column is problematic,
#        how can we add 10 to each value in that column?
ddf["Distance"].apply(lambda x: x + 10).compute()
```

TODO #10:

```
# TODO 10: The "Distance" column in the DataFrame is currently in
miles.
#         Our goal is to convert these units to kilometers.
# TODO 10.1 Create a function called `converter` that takes two
parameters: the DataFrame `df` and a multiplier.
#          The function should return the modified DataFrame with the
distance column updated
# TODO 10.2 Extract the "Distance" column and store the Series in a
parameter named `meta`.
# TODO 10.3 Use `map_partitions` to apply this function to each
internal pandas DataFrame in parallel.
# Hints: To convert 100 miles to kilometers, multiply by the
conversion factor 1.60934:
#        100 miles × 1.60934 = 160.934 kilometers

import pandas as pd

def converter(df, multiplier=1):
  return df * multiplier

meta = pd.Series(name="Distance", dtype="float64")
distance_km = ddf.Distance.map_partitions(
    converter, multiplier=0.6, meta=meta
)

distance_km.head()
```